Rešenja zadataka za kolokvijum iz Operativnih sistema 1 jun 2020.

1. (10 poena)

```
sys_call:   ; Save the current context
            push  r0    ; save r0 temporarily on the (kernel) stack
            load  r0, [running] ; r0 now points to the running PCB
            store r1, #offsR1[r0] ; save r1
            store r2, #offsR2[r0] ; save r2
            ...                   ; save r3-r30
            store r31, #offsR31[r0] ; save r31
            pop   r1                ; save r0
            push  r1
            store r1, #offsR0[r0]
            store SPC, #offsPC[r0]  ; save PC
            store SPSW,#offsPSW[r0] ; save PSW
            store SP, #offsSP[r0]   ; save SP
            load  r1, #offstR1[r0]  ; restore sys call args in r1
            pop   r0                ; and r0

            ; Execute the system call
            call  s_call

            ; Restore the new context
            load  r0, [running]
            load SP, #offsSP[r0] ; restore SP
            load SPSW,#offsPSW[r0] ; restore PSW
            load SPC, #offsPC[r0] ; restore PC
            load r31, #offsR31[r0] ; restore R31
            ...                    ; restore r30-r2
            load r1, #offsR1[r0] ; restore R1
            load r0, #offsR0[r0] ; restore R0

            ; Return
            iret
```

2. (10 poena)

```
void mat_add () {
  Thread* thr[M];
  Semaphore* sem[M];

  for (int i=0; i<M; i++) {
    sem[i] = new Semaphore(0);
    thr[i] = new RowAdder(mat[i],N,&res[i],sem[i]);
    thr[i]->start();
  }

  for (int i=0; i<M; i++) {
    sem[i]->wait();
    delete sem[i];
    delete thr[i];
  }
}
```

3. (10 poena) a)(3) VA(43): Page1(15):Page2(16):Offset(12).

b)(3) Kod pogrešnog pristupa elementu iza kraja svakog reda osim poslednjeg (tj. za sve `i<M-1` i `j=N`), proces pristupa prvom elementu sledećeg reda istog niza, tako da ti pristupi ne predstavaljaju prestup. Kod prekoračenja poslednjeg reda niza `src` (`i=M-1` i `j=N`), proces zapravo pristupa prvom elementu prvog reda niza `dst`, tako da ni to nije prestup. Međutim, kod upisa tog elementa u niz `dst`, proces će izazvati straničnu grešku (*page fault*) koju će operativni sistem tumačiti kao prestup, jer adresira stranicu koja ne pripada nekom alociranom segmentu. Ovo se dešava za `i=3` i `j=N`.

c)(4) Ovaj proces tokom izvršavanja regularno adresira sledeće stranice:

- jednu stranicu segmenta za kod, za dohvatanje instrukcija programa
- samo jednu stranicu segmenta za stek, jer je to dovoljno za samo jedan poziv potprograma `main` sa argumentima ove funkcije i njene dve automatske celobrojne promenljive na steku
- po 16 celih stranica za svaki niz `src` i `dst`, jer svaki niz sadrži $4*0x1000=4*2^{12}$ elemenata po 4 bajta, odnosno 2^{16} bajtova, što je po 16 stranica po 2^{12} bajtova.

Sve ukupno, proces regularno adresira 34 stranice. Kako svaka od njih ostaje u memoriji nakon prvog adresiranja i eventaulnog učitavanja, i pošto je procesu dodeljeno dovoljno okvira, ovaj proces će generisati isto toliki broj (34) straničnih grešaka koje će biti uspešno obrađene.

4. (10 poena)

```
#define max(a,b) ((a)>=(b)?(a):(b))

ulong allocateBlock (ulong startingFrom) {
  if (startingFrom>=NumOfBlocks) return 0;
  ulong bt = 0, blk = 0; byte msk = 0;
  ulong limit = max(NumOfBlocks-startingFrom,startingFrom);
  for (ulong i=0; i<limit; i++) {
    if (i<NumOfBlocks-startingFrom) {
      blk = startingFrom+i;
      blockToBit(blk,bt,msk);
      if ((blocks[bt]&msk)==0) {
        blocks[bt] |= msk;
        return blk;
      }
    }
    if (i<startingFrom && i>0) {
      blk = startingFrom-i;
      blockToBit(blk,bt,msk);
      if ((blocks[bt]&msk)==0) {
        blocks[bt] |= msk;
        return blk;
      }
    }
  }
  return 0;
}
```